Abstract

The University of Cambridge submission to the WMT18 news translation task focuses on the combination of diverse models of translation. We compare recurrent, convolutional, and self-attention-based neural models on German-English, English-German, and Chinese-English. Our final system combines all neural models together with a phrase-based SMT system in an MBR-based scheme. We report small but consistent gains on top of strong Transformer ensembles.

1 Introduction

Encoder-decoder networks (Pollack, 1990; Chrisman, 1991; Forcada and Ñeco, 1997; Kalchbrenner and Blunsom, 2013) are the current prevailing architecture for neural machine translation (NMT). Various architectures have been used in the general framework of encoder and decoder networks such as recursive auto-encoders (Pollack, 1990; Socher et al., 2011; Li et al., 2013), (attentional) recurrent models (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Wu et al., 2016; Chen et al., 2018), convolutional models (Kalchbrenner and Blunsom, 2013; Kaiser et al., 2017; Gehring et al., 2017), and, most recently, purely (self-)attention-based models (Vaswani et al., 2017; Ahmed et al., 2017; Shaw et al., 2018). In the spirit of Chen et al. (2018) we devoted our WMT18 submission to exploring the three most commonly used architectures: recurrent, convolutional, and self-attention-based models like the Transformer (Vaswani et al., 2017). Our experiments suggest that self-attention is the superior architecture on the tested language pairs, but it can still benefit from model combination with the other two. We show that using large batch sizes is crucial to Transformer training, and that the delayed SGD updates technique (Saunders et al., 2018) is useful to increase the batch size on limited GPU hardware. Furthermore, we also report gains from MBR-based combination with a phrase-based SMT system. We found this particularly striking as the SMT baselines are often more than 10 BLEU points below our strongest neural models. Our final submission ranks second in terms of BLEU score in the WMT18 evaluation campaign on English-German and German-English, and outperforms all other systems on a variety of linguistic phenomena on German-English (Avramidis et al., 2018).

2 System Combination

Stahlberg et al. (2017a) combined SMT and NMT in a hybrid system with a minimum Bayes-risk (MBR) formulation which has been proven useful even for practical industry-level MT (Iglesias et al., 2018). Our system combination scheme is a generalization of this approach to more than two systems. Suppose we want to combine $q$ models $M_1, \ldots, M_q$. We first divide the models into two groups by selecting a $p$ with $1 \leq p \leq q$. We refer to scores from the first group $M_1, \ldots, M_p$ as full posterior scores and from the second group $M_{p+1}, \ldots, M_q$ as MBR-based scores. Full posterior models contribute to the combined score with their complete posterior of the full translation. In contrast, models in the second group only provide the evidence space for estimating the probability of $n$-grams occurring in the translation. Full-posterior models need to assign scores via the standard left-to-right factorization of neural sequence models:

$$\log P(y_T^T|x, \mathcal{M}_i) = \sum_{t=1}^{T} \log P(y_t|y_t^{t-1}, x, \mathcal{M}_i)$$

(1)

for a target sentence $y = y_T^T$ of length $T$ given a source sentence $x$ for all $i \leq p$. For exam-
ple, all left-to-right neural models in this work can be used as full posterior models, but the right-to-left models (Sec. 3) and SMT cannot. We combine full-posterior scores log-linearly, and bias the combined score $S(y|x)$ towards low-risk hypotheses with respect to the MBR-based group as suggested by Stahlberg et al. (2017a, Eq. 4):

$$S(y|x) = \sum_{t=1}^{T} \left( \sum_{i=1}^{p} \lambda_i \log P(y_t|y_{t-1}, x, M_i) + \right) \text{Full posterior}$$

$$+ \sum_{j=p+1}^{q} \lambda_j \sum_{n=1}^{4} P(y_{t-n}^{j}|x, M_j) \right) \text{MBR-based n-gram scores}$$

(2)

where $\lambda_1, \ldots, \lambda_q$ are interpolation weights. Eq. 2 also describes how to use beam search in this framework as hypotheses can be built up from left to right due to the outer sum over time steps. The MBR-based models contribute via the probability $P(y_{t-n}^j|x, M_j)$ of an $n$-gram $y_{t-n}^j$ given the source sentence $x$. Posteriors in this form are commonly used for MBR decoding in SMT (Kumar and Byrne, 2004; Tromble et al., 2008), and can be extracted efficiently from translation lattices using counting transducers (Blackwood et al., 2010).

For our neural models we run beam search with beam size 15 and compute posteriors over the 15-best list. We smooth all $n$-gram posteriors as suggested by Stahlberg et al. (2017a).

Note that our generalization to more than two systems can still be seen as instance of the original scheme from Stahlberg et al. (2017a) by viewing the first group $M_1, \ldots, M_p$ as ensemble and the evidence space from the second group $M_{p+1}, \ldots, M_q$ as mixture model.

The performance of our system combinations depends on the correct calibration of the interpolation weights $\lambda_1, \ldots, \lambda_q$. We first tried to use $n$-best or lattice MERT (Och, 2003; Macherey et al., 2008) to find interpolation weights, but these techniques were not effective in our setting, possibly due to the lack of diversity and depth in $n$-best lists from standard beam search. Therefore, we tune on the first best translation using Powell’s method (Powell, 1964) with a line search algorithm similar to golden-section search (Kiefer, 1953).

## 3 Right-to-left Translation Models

Standard NMT models generate the translation from left to right on the target side. Recent work has shown that incorporating models which generate the target sentence in reverse order (i.e. from right to left) can improve translation quality (Liu et al., 2016; Li et al., 2017; Sennrich et al., 2017; Hassan et al., 2018). Right-to-left models are often used to rescore $n$-best lists from left-to-right models. However, we could not find improvements from resoring in our setting. Instead, we extract $n$-gram posteriors from the R2L model, reverse them, and use them for system combination as described in Sec. 2.

## 4 Experimental Setup

### 4.1 Data Selection

We ran language detection (Nakatani, 2010) and gentle length filtering based on the number of characters and words in a sentence on all available monolingual and parallel data in English, German, and Chinese. Due to the high level of noise in the ParaCrawl corpus and its large size compared to the rest of the English-German data we additionally filtered ParaCrawl more aggressively with the following rules:

- No words contain more than 40 characters.
- Sentences must not contain HTML tags.
- The minimum sentence length is 4 words.
- The character ratio between source and target must not exceed 1:3 or 3:1.
- Source and target sentences must be equal after stripping out non-numerical characters.
- Sentences must end with punctuation marks.

This additional filtering reduced the size of ParaCrawl from originally 36M sentences to 19M sentences after language detection, and to 11M sentences after applying the more aggressive rules.

For backtranslation (Sennrich et al., 2016a) we selected 20M sentences from News Crawl 2017. We used a single Transformer (Vaswani et al., 2017) model in Tensor2Tensor’s (Vaswani et al., 2018) transformer base configuration...
Table 1: Training data sizes for English-German and German-English after filtering.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Over-sampling</th>
<th>#Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Crawl</td>
<td>2x</td>
<td>4.43M</td>
</tr>
<tr>
<td>Europarl v7</td>
<td>2x</td>
<td>3.76M</td>
</tr>
<tr>
<td>News Commentary v13</td>
<td>2x</td>
<td>0.57M</td>
</tr>
<tr>
<td>Rapid 2016</td>
<td>2x</td>
<td>2.27M</td>
</tr>
<tr>
<td>ParaCrawl</td>
<td>1x</td>
<td>11.16M</td>
</tr>
<tr>
<td>Synthetic (news-2017)</td>
<td>1x</td>
<td>20.00M</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>42.19M</strong></td>
</tr>
</tbody>
</table>

Table 2: Training data sizes for Chinese-English after filtering.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Over-sampling</th>
<th>#Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWMT - CASIA2015</td>
<td>2x</td>
<td>2.08M</td>
</tr>
<tr>
<td>CWMT - CASICT2015</td>
<td>2x</td>
<td>3.95M</td>
</tr>
<tr>
<td>CWMT - Datum2017</td>
<td>2x</td>
<td>1.93M</td>
</tr>
<tr>
<td>CWMT - NEU2017</td>
<td>2x</td>
<td>3.95M</td>
</tr>
<tr>
<td>News Commentary v13</td>
<td>2x</td>
<td>0.49M</td>
</tr>
<tr>
<td>UN v1.0</td>
<td>1x</td>
<td>14.25M</td>
</tr>
<tr>
<td>Synthetic (news-2017)</td>
<td>1x</td>
<td>20.00M</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>46.66M</strong></td>
</tr>
</tbody>
</table>

Table 3: Number of model parameters.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>en-de</th>
<th>de-en</th>
<th>zh-en</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>114.2M</td>
<td>192.7M</td>
<td></td>
</tr>
<tr>
<td>SliceNet</td>
<td>27.5M</td>
<td>86.4M</td>
<td></td>
</tr>
<tr>
<td>Transformer</td>
<td>212.8M</td>
<td>291.4M</td>
<td></td>
</tr>
<tr>
<td>Relative Transformer</td>
<td>213.8M</td>
<td>292.5M</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Impact of the effective batch size on Transformer training on en-de news-test2017 after 3,276M training tokens, beam size 4.

<table>
<thead>
<tr>
<th>#Physical GPUs (g)</th>
<th>Delay factor (d)</th>
<th>#Effective GPUs (g'=gd)</th>
<th>Effective batch size (b'=bg')</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2.048</td>
<td>28.2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>8.192</td>
<td>29.5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>16</td>
<td>32.768</td>
<td>30.3</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>64</td>
<td>131.072</td>
<td>29.8</td>
</tr>
</tbody>
</table>

English-German and German-English, but not on Chinese-English.

**LSTM** For our recurrent models we adapted the TensorFlow seq2seq tutorial code base (Luong et al., 2017) for use inside the Tensor2Tensor library (Vaswani et al., 2018). We roughly followed the UEdin WMT17 submission (Sennrich et al., 2017) and stacked four 1024-dimensional LSTM layers with layer normalization (Ba et al., 2016) and residual connections in both the decoder and bidirectional encoder. We equipped the decoder network with Bahdanau-style (Bahdanau et al., 2015) attention (normed_bahdanau).

**SliceNet** The convolutional model of Kaiser et al. (2017) called SliceNet is implemented in Tensor2Tensor. We use the standard configuration slicenet_1 of four hidden layers with layer normalization.

**Transformer** We compare two Transformer variants available in Tensor2Tensor: the original Transformer (Vaswani et al., 2017) (transformer_big setup) and the Transformer of Shaw et al. (2018) with relative positional embeddings (transformer_relative_big setup). Both use 16-head dot-product attention and six 1024-dimensional encoder and decoder layers.

The number of training parameters of our neural models is summarized in Tab. 3.

---

2. https://github.com/fxsjy/jieba
3. https://github.com/fstahlberg/tensor2tensor-usr
4.4 Training

We train vanilla phrase-based SMT systems\(^5\) and extract 1000-best lists of unique translations candidates, from which \(n\)-gram posteriors are calculated.

All neural models were trained with the Adam optimizer (Kingma and Ba, 2015), dropout (Srivastava et al., 2014), and label smoothing (Szegedy et al., 2016) using the Tensor2Tensor (Vaswani et al., 2018) library. We decode with the average of the last 40 checkpoints (Junczys-Dowmunt et al., 2016a).

We make extensive use of the delayed SGD updates technique we already applied successfully to syntax-based NMT (Saunders et al., 2018). Delaying SGD updates allows to arbitrarily choose the effective batch size even on limited GPU hardware. Large batch training has received some attention in recent research (Smith et al., 2017; Neishi et al., 2017) and has been shown particularly useful for training the Transformer architecture with the Tensor2Tensor framework (Popel and Bojar, 2018). We support these findings in Tab. 4.\(^6\) Our technical infrastructure\(^7\) allows us to train on four P100 GPUs simultaneously, which limits the number of physical GPUs to \(g = 4\) and the batch size\(^8\) to \(b = 2048\) due to the GPU memory. Thus, the maximum possible effective batch size without delaying SGD updates is \(b' = 8192\). Training with delay factor \(d\) accumulates gradients over \(d\) batches and applies the optimizer update rule on the accumulated gradients. This allows us to scale up the effective number of GPUs to 16 and improve the BLEU score significantly (29.5 vs. 30.3). Note that training regimens are equivalent if their effective batch size is the same, i.e. training on 4 physical GPUs with \(d = 4\) is mathe-

\(^5\)Excluding the UN corpus and the backtranslated data.
\(^6\)We had to reduce the learning rate for \(g' = 1\) to avoid training divergence.
\(^7\)http://www.hpc.cam.ac.uk/
\(^8\)We follow Vaswani et al. (2017, 2018) and specify the batch size in terms of number of source and target tokens in a batch, not the number of sentences.

### Table 5: Training setups for our neural models on all language pairs.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>#Effective GPUs</th>
<th>Batch size</th>
<th>#SGD updates</th>
<th>#Training tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>8</td>
<td>4,096</td>
<td>45K</td>
<td>1.475M</td>
</tr>
<tr>
<td>SliceNet</td>
<td>4</td>
<td>2,048</td>
<td>800K</td>
<td>6.554M</td>
</tr>
<tr>
<td>R2L Transformer</td>
<td>16</td>
<td>2,048</td>
<td>200K</td>
<td>6.554M</td>
</tr>
<tr>
<td>Transformer</td>
<td>16</td>
<td>2,048</td>
<td>250K</td>
<td>8.192M</td>
</tr>
<tr>
<td>Relative Transformer</td>
<td>16</td>
<td>2,048</td>
<td>250K</td>
<td>8.192M</td>
</tr>
</tbody>
</table>

\[\text{Table 5: Training setups for our neural models on all language pairs.}\]

4.5 Decoding

We use the beam search strategy with beam size 8 of the SGNMT decoder (Stahlberg et al., 2017b, 2018) in all our experiments. We apply length normalization (Bahdanau et al., 2015) on German-English and Chinese-English but not on English-German. As outlined in Sec. 2 we either use full posteriors or MBR-style \(n\)-gram posteriors from our individual models. SMT \(n\)-gram scores are extracted as described by Blackwood et al. (2010) using HiFST’s lmbr tool. We use SGNMT’s ngram output format to extract \(n\)-gram scores from our neural models.

5 Results

On English-German and German-English news-test2014 we compute cased BLEU scores with Moses’ multi-bleu.pl script on tokenized output to be comparable with prior work (Wu et al., 2016; Kaiser et al., 2017; Gehring et al., 2017; Vaswani et al., 2017; Chen et al., 2018). On all other test sets we use mteval-v13a.pl to be comparable to the official cased WMT scores.\(^9\)

First, we will discuss our experiments with a single architecture, i.e. single systems and ensembles of two systems with the same architecture. Tab. 6 compares the architectures on all test sets. PBMT as a single system is clearly inferior to all neural systems. Ensembling neural systems helps for all architectures across the board. LSTM

\[^9\]http://matrix.statmt.org/
is usually slightly better than the convolutional Slicenets, but is much slower to train and decode (cf. Tab. 3). Note that our LSTM 2-ensemble is on par with the best BLEU score in WMT17 (Sennrich et al., 2017), which was also based on recurrent models. Transformer architectures outperform LSTMs and Slicenets on all test sets. The right-to-left Transformer is usually slightly worse, the Transformer with relative positioning slightly better than the standard Transformer setup.

Tab. 7 summarizes our system combination results with multiple architectures. Adding LSTM and Slicenets as full-posterior models to an ensemble of a Transformer and a Relative Transformer does not improve the BLEU score (rows 6 vs. 7). We see very slight improvements when we use these models to extract \( n \)-gram scores instead (rows 6 vs. 8). We report further gains by using MBR-based \( n \)-gram scores from the right-to-left Transformer and the PBMT system. The improvements from adding PBMT are rather small, but we still found them surprising given that the PBMT baseline is usually more than 10 BLEU points worse than our best single neural model. We list the performance of our submitted systems on all test sets in Tab. 8.

### 6 Related Work

There is a large body of research comparing NMT and SMT (Schnober et al., 2016; Toral and Sánchez-Cartagena, 2017; Koehn and Knowles, 2017; Menacer et al., 2017; Dowling et al., 2018; Bentivogli et al., 2016, 2018). Most studies have found superior overall translation quality of NMT models in most settings, but complementary strengths of both paradigms. Therefore, the literature about hybrid NMT-SMT sys-
tems is also vast, ranging from rescoring and reranking methods (Neubig et al., 2015; Stahlberg et al., 2016; Khayrallah et al., 2017; Grundkiewicz and Junczys-Dowmunt, 2018; Avramidis et al., 2016; Marie and Fujita, 2018), MBR-based formalisms (Stahlberg et al., 2017a, 2018; Iglesias et al., 2018), NMT assisting SMT (Junczys-Dowmunt et al., 2016b; Du and Way, 2017), and SMT assisting NMT (Niehues et al., 2016; He et al., 2016; Long et al., 2016; Wang et al., 2017; Dahlmann et al., 2017; Zhou et al., 2017). We confirm the potential of hybrid systems by reporting gains on top of very strong neural ensembles.

Ensembling is a well-known technique in NMT to improve system performance. However, ensembles usually consist of multiple models of the same architecture. In this paper, we compare and combine three very different architectures (recurrent, convolutional, and self-attention based) in two different ways (full posterior and MBR-based), and find that combination with MBR-based n-gram scores is superior.

7 Conclusion

We have described our WMT18 submission, which achieves very competitive BLEU scores on all three language pairs (English-German, German-English, and Chinese-English) and significantly higher accuracies in a variety of linguistic phenomena compared to other submissions (Avramidis et al., 2018). Our system combines three different neural architecture with a traditional PBMT system. We showed that our MBR-based scheme is effective to combine these diverse models of translation, and that adding the PBMT system to the mix of neural models still yields gains although it is much worse as stand-alone system.

Acknowledgments

This work was supported in part by the U.K. Engineering and Physical Sciences Research Council (EPSRC grant EP/L027623/1).

References


Shuyo Nakatani. 2010. Language detection library for Java.


